Claims

1. A method of processing requests for the allocation of a memory block of a data memory, wherein segments of the data memory are allocated to different levels according to their size, the method comprising the steps of:

(a) receiving a request for the allocation of a memory block;

(b) determining the lowest of said levels containing a segment of the same size as or larger than the requested memory block;

(c) determining, in the level determined in step (b), the availability of a free segment of a size the same as or larger than the requested memory block; and

(d) depending on the determination in step (c), allocating a free segment.

2. The method of claim 1; further comprising:

(e) repeating steps (c) and (d) for the next higher level if no free segment of a size the same as or larger than the requested memory block has been found in step (c); and

(f) repeating step (e) until a free segment has been allocated or there is no next level.

3. The method of claim 1 or 2, wherein each level is associated with a different granule size to the power of two, and the sizes of memory segments allocated to a level are related to the granule size of the respective level.

4. The method of claim 3, wherein the granule size associated with a level defines the size difference between memory segments allocated to that level.

5. The method of claim 4, wherein step (a) further comprises rounding the requested memory block to the lowest granule size before performing steps (b) to (d).

6. The method of any preceding claims, wherein each level is associated with a table of pointers indicative of memory addresses of free memory segments of a size allocated to the respective level.

7. The method of any preceding claim, wherein step (d) comprises returning a pointer to the allocated free segment.

8. The method of any preceding claim, wherein (d) comprises returning a null pointer if no free segment is allocated.

9. The method of any preceding claim, wherein a bitmap is indicative of the state of memory segments (free, allocated), the bitmap comprising a root bitmap, each bit of the root bitmap being indicative of whether or not an associated one of said levels contains at least one free segment, and wherein step (b) further comprises determining from the root bitmap said lowest level containing a segment of a size the same as or larger than the requested memory block

10. The method of any preceding claim, wherein step (a) comprises receiving a binary data set indicative of the size of the requested memory block, wherein each bit of the binary data set is associated with an entry of a lookup table associated with one of said levels, and step (b) comprises determining the most significant set bit of the binary data set, and determining from the entry of the lookup table associated with the most significant set bit the lowest of said levels containing a segment of a size the same as or larger than the requested memory block.

11. The method of claims 9 and 10, wherein each mask of a set of predetermined masks is associated with one of said levels, and step (c) further comprises performing a logic operation on the mask associated with the lowest level determined in step (b) and said binary data set, wherein the operation result is an index to bits of the bitmap indicative of the state of a segment of a size the same as or larger than the requested memory block.

12. The method of claim 11, wherein said bitmap comprises a plurality of second and third stage bitmaps, each bit of the root bitmap being indicative of the state of the bits of an associated one of said second stage bitmaps, each bit of said second stage bitmaps being indicative of the state of an associated predetermined number of bits of one of said third stage bitmaps, and each bit of the third stage bitmap being indicative of whether or not an associated segment is free, and wherein the operation result is an index to one bit of the second stage bitmap and one bit of said predetermined number of bits of the third stage bitmap associated with said one bit of the second stage bitmap, said one bit of the third stage bitmap

being indicative of the state of a segment of a size the same as or larger than the requested memory block.

13. The method of claim 12, wherein step (c) further comprises, if no free segment is found, repeating the determination for the next more significant bit of said predetermined number of bits of the third stage bitmap, until a free segment is found or there is no more significant bit of said predetermined number of bits of the third stage bitmap.

14. The method of claim 13, wherein step (c) further comprises, if no free segment is found, repeating the determination for the predetermined number of bits of the third stage bitmap associated with the next more significant set bit of the second stage bitmap, until a free segment is found or there is no more significant bit of said one second stage bitmap.

15. The method of claim 14, wherein step (c) further comprises, if no free segment is found, repeating the determination for the second stage bitmap associated with the next more significant set bit of the root bitmap, until a free segment is found or there is no more significant bit of the root bitmap.

16. The method of claims 12 and 15, wherein each bit of the third stage bitmaps is associated with an entry in a table of pointers indicative of memory addresses of free memory segments.

17. A method of managing a data memory, the method comprising:

defining a number of levels of the data memory;

defining a different granule size for each level;

defining a different range of a plurality of different sizes of memory segments for each level, wherein the size of each memory segment is related to the granule size of the respective level, and wherein a request for the allocation of a memory block is processable by determining a level containing segments of the same size as or larger than the requested memory block, and allocating a free segment of a size the same as or larger than the requested memory block in that level.

18. The method of claim 17, wherein the granule size defines the size difference between memory segments in each level.

19. The method of claim 17 or 18, further comprising:

generating a bitmap indicative of the state of each segment (free, allocated) and of whether or not a level contains at least one free segment.

20. The method of claim 19, wherein the bitmap comprises a root bitmap, each level being associated with one bit of the root-bitmap, and a plurality of second and third stage bitmaps associated with the segments, each bit of the root bitmap being indicative of the state of the bits of an associated one of said second stage bitmaps, and each bit of said second stage bitmaps being indicative of the state of an associated predetermined number of bits of one of said third stage bitmaps.

21. The method of claim 19 or 20, further comprising:

updating the bitmap when a segment is allocated.

22. The method of claim 19, 20 or 21, further comprising:

updating the bitmap when a segment is freed.

23. The method of any of claims 17 to 22, further comprising generating a table of pointers for each level indicative of memory addresses of free memory segments of a size associated with the respective level.

24. The method of claim 23 as dependent on any of claims 19 to 22, wherein each bit of the third stage bitmaps is associated with an entry in the tables of pointers.

25. The method of any of claims 17 to 24, further comprising:

generating a lookup table, wherein each entry of the lookup table is associated with a bit of a binary data set indicative of the size of the requested memory block and indicative of one of said levels.

26. The method of any of claims 17 to 25, further comprising:

generating a set of masks, wherein each of the set of masks is associated with one of said levels, and wherein a logical operation of a binary data set indicative of the size of the requested memory block and the mask associated with a

level containing segments of the same size as or larger than the requested memory block results in an index to a segment of a size the same as or larger than the requested memory block in that level.

27. A method of managing a data memory comprising memory segments of different sizes for allocation in response to a memory allocation request, the method comprising:

creating a first doubly linked list of consecutive memory segments irrespective of size and status (free, allocated); and

creating a second doubly linked list of free memory segments of the same size.

28. The method of claim 27, wherein memory segments in the first doubly linked list are arranged in the order of associated memory addresses.

29. The method of claim 27 or 28, further comprising, when freeing a memory segment:

determining the state of memory segments adjacent to the memory segment to be freed using the first doubly linked list;

merging the memory segment to be freed with free adjacent memory segments; and

updating the first and second doubly linked lists accordingly.

30. The method of any of claims 27 to 29, wherein each second doubly linked list is a LIFO (Last In First Out) list.

31. The method of any of claims 27 to 30, comprising updating the second doubly linked list upon allocation of a memory segment upon request.

32. The method of any of claims 27 to 31, further comprising, if a segment determined for allocation upon request is larger than a requested memory block:

allocating a portion of the determined segment large enough to satisfy the request;

providing the remaining portion as a new free memory segment; and

updating the first and second doubly linked lists accordingly.

33. The method of any of any of claims 27 to 32, wherein each segment is associated with a header, thereby to form the first doubly linked list, each header including information indicative of the size of the associated segment, information indicative of the state (free, allocated) of the associated segment, and a pointer indicative of the memory address of the previous segment.

34. The method of claims 30 and 33, wherein the header associated with each free segment of a given size further includes a pointer indicative of the memory address of a previous and/or subsequent free segment of the same size, depending on the availability of a previous and/or subsequent free segment of the same size and in accordance with the order of free segments of the same size in the LIFO list.

35. A method of managing a data memory, the method comprising:

allocating free segments of the data memory to different levels according to their size; and

providing a bitmap comprising different stages, wherein the bits of one stage are indicative of the availability of free segments in said levels, and the bits of another stage are indicative of the state and/or size and/or location of individual segments.

36. The method of claim 35, wherein the bits of one stage are associated with pointers indicative of the memory address of free segments.

37. The method of claim 35 or 36, further comprising:

updating the bitmap to reflect the allocation or release of memory segments.

38. A method of managing a data memory, including freeing and allocating segments of the data memory, the method comprising, when freeing a memory segment:

determining the state of memory segments adjacent to the memory segment to be freed; and

merging the memory segment to be freed with free adjacent memory segments.

39. A method of managing a data memory, including the method of any of claims 17 to 26 and/or claims 27 to 34 and/or claims 35 to 37 and/or claim 38.

40. An operating system for a computer, adapted to perform the method of any preceding claim.

41. The operating system of claim 40, wherein the operating system is a realtime operating system.

42. The operating system of claim 40 or 41, adapted to perform the method of any of claims 1 to 39 at task level.

43. The operating system of any of claims 40 to 42, adapted to perform the method of any of claims 1 to 39 at interrupt level.

44. A computer program adapted to perform the method of any of claims 1 to 39 when operated on a computer.

45. A storage medium having stored thereon a set of instructions, which when executed by a computer, performs the method of any of claims 1 to 39.

46. A computer system programmed to perform the method of any of claims 1 to 39.

47. A processor arranged to perform the method of any of claims 1 to 39.